OULUN SEUDUN
AMMATTIKORKEAKOULU

Jouni Suorsa

VEDENKORKEUS-KOMPONENTTI

VEDENKORKEUS-KOMPONENTTI

Jouni Suorsa
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistojen kehitys

Tekijä: Jouni Suorsa
Opinnäytetyön nimi: Vedenkorkeus-komponentti
Työn ohjaaja: Markku Rahikainen
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 31

Oulun Seudun Melojat ry:llä on ollut sivuillaan vedenkorkeustietoja vaikealukuisina kuvina ja heillä on ollut tarve komponentille, joka näyttäisi käyttäjille vedenkorkeustietoja yksinkertaisessa muodossa.

Opinnäytetyön tarkoituksena oli luoda melojille melontaseuran verkkopalvelimella sijaitseva vedenkorkeutta ilmaiseva palvelu. Palvelu lukee vedenkorkeuden ympäristöhallinnon palvelun kautta melontapaikoille ja esittää tiedon melojille yksinkertaisesti ja helposti ymmärrettävästi.

Melontapaikat on ennalta syötetty järjestelmään, ja niille merkitään vedenkorkeuden rajat, milloin ne vielä kelpaavat melottaviksi. Vedenkorkeus haetaan ympäristöhallinnon palvelusta kerran vuorokaudessa.

Komponentin kehittämiseen käytettiin PHP-ohjelmointikieltä ja kehitysalustana Joomla 2.5 -sisällönhallintajärjestelmää.

Opinnäytetyön aikana saatiin toteutettua komponentti, joka hakee tiedot ympäristöhallinnon palvelun kautta, tallentaa tiedot tietokantaan ja esittää ne sivuston käyttäjälle mahdollisimman yksinkertaisessa muodossa. Ylläpitoa helpottamaan tehtiin ylläpitosivut, joiden kautta melontapaikkojen tietoja ja asetuksia pääsee muokkaamaan. Toteutettu komponentti vastasi toimeksiantajan vaatimuksia.

Asiasanat: PHP, MySQL, Joomla, visualisointi

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Programming

Author: Jouni Suorsa
Title of thesis: River Information Component
Supervisor: Markku Rahikainen
Term and year when the thesis was submitted: Spring 2013 Pages: 31

Oulun Seudun Melojat (OSM) is a river touring and paddling club, which has had river height information on their website in a hard-to-read image format. They have had the need for a component, which would show river information in a more compact easy-to-read format.

The target of this thesis was to create a component to be used on the club's web-server. The component needed to be able to fetch river information from Finland's environmental administration's service, and to display that information on the club's website in a very easy to understand format.

Rivers on the component are pre-configured to the settings, and height-limits are set, which indicate when the river is paddle able.

The component was developed for Joomla Content Management System with the PHP programming language.

The final product of this thesis was a component that fetched information from the environmental administration's service, saved the information to a database, and displayed it on the club's website in a very easy to read format. To ease administration, administration-pages were also developed, when information and settings are easily editable. The component met with all the client's needs and demands.

Keywords: PHP, MySQL, Joomla, visualization

ALKULAUSE

Oulun Seudun Melojat ry:n Internet-sivuista vastaava Timo Kallio otti yhteyttä Oulun seudun ammattikorkeakouluun vedenkorkeusjärjestelmän toteuttamisesta yhdistyksen sivustolle. Kuultuani opinnäytetyöaiheesta otin yhteyttä Timoon ja tammikuussa 2013 keskustelimme työn toteuttamisesta. Ideana oli lähteä toteuttamaan vedenkorkeusjärjestelmää Joomla-sisällönhallintajärjestelmän lisäosana. Komponentti hakee vedenkorkeustiedot ympäristöhallinnon sivulta ja esittää käyttäjälle yksinkertaisessa muodossa, onko melontapaikka melontakelpoinen.

Norjassa hieman samantyylinen sivusto on kahden norjalaisen henkilön, Nils Tarjei Hjelmen ja Kristian Waldalin, ylläpitämä Riverflow.no-sivusto osoitteessa http://www.riverflow.no.

Haluan kiittää Oulun Seudun Melojat ry:tä, erityisesti Timo Kalliota, jolta sain aiheen opinnäytetyölle. Kiitokset menevät myös työn ohjaajille ja kaikille, jotka kannustivat minua työn tekemisen ajan.

Toukokuussa 2013

Jouni Suorsa

SISÄLLYS

SANASTO

Funktio	Aliohjelma ohjelmassa tai luokassa
Luokka	Olion tyypin määritelmä
Olio	Tietoja ja toimintoja sisältävä ohjelmoinnin käsite
Parserointi	Merkkijonon käsittely käytettävään muotoon
PHP	PHP: Hypertext Preprocessor – ohjelmointikieli
Rajapinta	Ulkoisesti paljastettu toimintojen käytettävyys
SQL	Structured Query Language – kyselykieli
WWW	World Wide Web

1 JOHDANTO

Oulun Seudun Melojat ry on retki- ja koskimelontaan erikoistunut seura, joka on toiminut jo vuodesta 1986. Oulun Seudun Melojat ry:n internet-sivuilla on ollut ennen opinnäytetyön toteuttamista muutaman kosken vedenkorkeustiedot, mutta ne on esitetty kuvaajina, joita on ollut työläs tulkita, eivätkä ne ole suoraan kertoneet, onko koski melomiskelpoinen.

Työssä lähdettiin toteuttamaan Oulun Seudun Melojat ry:n Joomla-sivustolle komponenttia, joka kertoo sivun käyttäjälle mahdollisimman yksinkertaistetusti melontapaikan melomiskelpoisuuden. Komponentti myös sisältää tarvittavien tietojen hakemisen verkon yli ympäristöhallinnon tarjoamasta palvelusta ja ylläpitoa helpottavan ylläpitokäyttöliittymän, josta pääsee lisäämään ja muokkaamaan melontapaikkoja tietokantaan.

Komponentti helpottaa melontapaikkojen melontakelpoisuuden tulkintaa, koska nykyinen graafinen korkeusdiagrammi kertoo asiaan perehtymättömälle vain korkeuden eikä kerro, miten vedenkorkeus soveltuu melontaan. Toteutettu komponentti sen sijaan kertoo korkeuden lisäksi, onko mitattu korkeus soveltuva melomisharrastukseen.

Opinnäytetyön raportoinnista rajattiin pois kehityspalvelimen ja Joomla-sisällönhallinnan asentaminen.

2 TYÖN MÄÄRITTELY

Työssä toteutettiin komponentti Joomla-sisällönhallintajärjestelmään. Asiakkaan toiveesta komponenttiin toteutettiin ylläpito-osa ja vedenkorkeusosa. Asiakas toivoi, että ylläpito-osassa pääsisi lisäämään ja poistamaan melontapaikkoja, nimeämään syötettyjä paikkoja sekä määrittämään melontakelpoisuuden raja-arvot. Lisäksi toivottiin, että melontapaikan sijainnin voisi jollain tapaa määrittää.

Asiakas toivoi käyttäjälle näkyvään vedenkorkeusnäkymään listausta melontapaikoista, joissa näkyisi melontapaikan nimi, veden virtaama tai korkeus sekä graafinen ilmaisu siitä, onko melontapaikka melomiskelpoinen. Värikoodit olivat keltainen liian vähäiselle vesimäärälle, vihreä melontakelpoiselle vesimäärälle ja punainen liialliselle vesimäärälle.

Komponentin täytyi myös pystyä hakemaan vedenkorkeus- ja virtaamatiedot ympäristöhallinnon ylläpitämästä OIVA-palvelusta, jonka löytää osoitteesta http://wwwp2.ymparisto.fi/scripts/oiva.asp, ja tallentamaan ne tietokantaan.

3 KÄYTETYT KEHITYSYMPÄRISTÖT

Työssä käytetyt kehitysympäristöt määräytyivät toimeksiantajan olemassa olevien sivujen pohjalta.

3.1 PHP

PHP on laajalti käytössä oleva yleiskäyttöinen komentosarjakieli, joka soveltuu erityisen hyvin dynaamisten internet-sivujen kehittämiseen. PHP on helppo oppia, mutta se soveltuu myös edistyneelle ammattilaiselle. (1.) Kielessä on olio-ominaisuudet ja tuki useimmille tietokannoille ja käyttöjärjestelmille.

PHP on täysin palvelimella suoritettava, eli koodi käsitellään palvelimella ja suoritetun koodin perusteella luodaan HTML-sivu. Sivu lähetetään WWW-selaimelle, joten se ei vaadi mitään erikoista tukea WWW-selaimelta. (2.) PHP-koodi saadaan upotettua HTML-kielen sekaan niin, että PHP-esikäsittelijä ei tulkitse HTML-kieltä lainkaan, vaan PHP:lla toteutetut toiminnallisuudet tulkitaan ja tulostetaan HTML-kielen sekaan.

PHP on saatavilla useimmille käyttöjärjestelmille, ja se on tuettu useimmilla WWW-palvelinohjelmistoilla. PHP 5 on julkaistu vuonna 2004. Kehityspalvelimella PHP:sta oli käytössä versio 5.3.3. Oulun Seudun Melojat ry:n palvelimella on käytössä versio 5.3.19.

3.2 MySQL

MySQL on suosituin avoimeen lähdekoodiin perustuva relaatiotietokanta-järjestelmä. MySQL:n kehityksestä ja tuesta vastaa nykyisin Oracle Corporation. MySQL soveltuu niin pieniin ostoslistatyylisiin tietokantoihin kuin myös suurten yritysten massiivisiin tietokantoihin ja Facebookin tyylisiin maailmanluokan Internet-palveluihin. (3; 4.)

MySQL on saatavilla useimmille käyttöjärjestelmille, ja sen käyttämiseen on kirjastot useimmissa ohjelmointikielissä ja kehitysympäristöissä. MySQL:n uusin versio on MySQL 5.6.11. Kehityspalvelimella oli käytössä MySQL 5.1 -haaraa edustava 5.1.69.

3.3 Joomla

Joomla on ilmainen ja vapaaseen lähdekoodiin perustuva sisällönhallintajärjestelmä (5). Joomla asennetaan Internet-palvelimelle ja sitä käytetään ja hallitaan WWW-selaimen kautta. Sen avulla Internet-sivustojen ylläpitäjän on helppo hallita sivustonsa sisältöä.

Joomla on myös laajennettavissa lisäosilla. Joomlalle on olemassa tuhansittain valmiita lisäosia, ja se tarjoaa useita kehittämistä helpottavia ja nopeuttavia toimintoja ja luokkia.

Joomlan uusimmat versiohaarat, 3.1 ja 3.0, tukevat muutamaa tietokanta-järjestelmää, mutta kehityksessä käytetty 2.5 -haara tukee vain MySQL-tietokantajärjestelmää (6). Kehityksessä käytettiin vanhempaa versiota, jotta se vastaisi toimeksiantajan sivustolla käytettyä versio-haaraa.

4 JOOMLA-KOMPONENTTI

Komponenttiin toteutettiin tietojen haku OIVA-järjestelmästä, tiedon tallennus tietokantaan, ylläpitoa helpottava osa ja käyttäjälle näkyvät sivut.

4.1 Komponentin kehityksessä käytetyt työkalut

Kehityspalvelimena toimi CentOS 6.4 -Linux-distribuutio. Palvelimelle oli kehitysvaiheessa asennettuna Apache 2.2.15 -www-palvelin, MySQL 5.1.69 -tietokantapalvelin ja PHP 5.3.3 -komentokielituki. Kehityksen alkuvaiheessa palvelimelle asennettiin Joomla 2.5.9.

Kehitystä varten palvelimelle asennettiin Subversion-versionhallintajärjestelmän versio 1.6.11, ja työlle tehtiin säilö versionhallintajärjestelmään. PHP-ohjelmointi tehtiin SSH-etäyhteyden yli KiTTy -SSH-asiakasohjelmalla, käyttäen palvelimelle asennettua nano-tekstieditoria. Tietokantamuokkaukset tehtiin MySQL:n mysql-komentorivityökalulla.

OIVA-palvelua käytettiin Mozilla Firefox 20 -selaimella, ja selaimen ja palvelun välistä yhteydenpitoa tarkkailtiin Wireshark-sovelluksella. Komponentin testauksessa käytettiin pääasiassa Mozilla Firefox 20 -selainta. Komponentin toimivuus älypuhelimen kautta käytettäessä testattiin myös Android-mobiilikäyttöjärjestelmän Chrome-selaimella.

Raportointiin käytettiin LibreOffice Writer ja Microsoft Word 2013 -tekstinkäsittelysovellusia. Raportoinnissa käytetyt koodin kuvakaappaukset otettiin Notepad++-tekstieditorista ja tallennettiin Microsoft Paint -piirto-ohjelmalla.

4.2 Tiedon haku ja parserointi OIVA-palvelusta

Järjestelmään toteutettiin tietojen haku OIVA-palvelun kautta, jota varten tutkittiin palvelua, haun toteuttamista ja haettujen tuloksien käsittelyä.

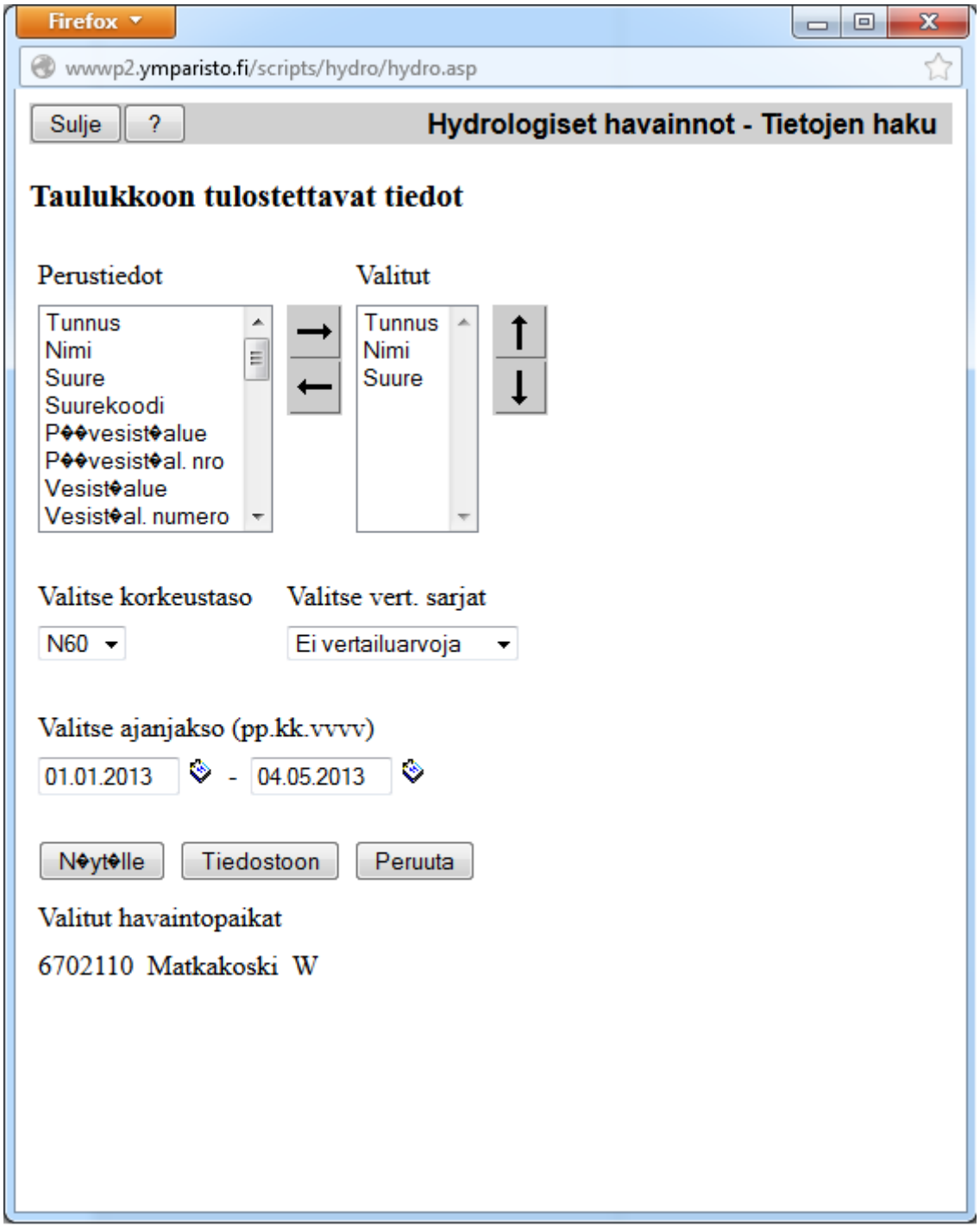
4.2.1 Tiedon haun tutkiminen

Työn alussa testattiin komponentin tärkeintä ominaisuutta, eli OIVA-tietojen hakua. Ei ollut täysin selvää, mitä kaikkea täytyy tehdä, että vedenkorkeustiedot saadaan haettua ympäristöhallinnon OIVA-palvelusta. Jos tietoa ei saada haettua ja käsiteltyä, on komponentti täysin turha toteuttaa.

Luotiin tunnus OIVA-palveluun ja kirjauduttiin sisään. OIVAssa on kaksi erillistä rajapintaa, Ympäristösuojelun tietojärjestelmä Vahti ja Ympäristötiedon hallintajärjestelmä Hertta. Vedenkorkeustiedot löytyvät Hertta-palvelun kautta, joten käytettiin ja tutkittiin vain Hertta-palvelua.

Hertta-palvelusta löytyy valtava määrä eri tietoa, mutta työn kannalta oleellisia ovat vain vedenkorkeus- ja virtaamatiedot. Oikeiden tietojen löytäminen palvelusta tuntui haasteelliselta, mutta Oulun Seudun Melojat ry:n Internet-sivuista vastaavan henkilön avustuksella oikeat kohdat löytyivät ja päästiin tutkimaan, kuinka tarvittavia tietoja pääsee hakemaan PHP-koodin kautta.

Tietojen kysely järjestelmästä tapahtuu normaalisti HTML-lomakkeen (kuva 1) kautta, joten ensimmäiseksi tutkittiin, mitä elementtejä palvelun HTML-lomake sisältää ja mitä niistä täytyy lähettää palvelimelle tietojen kyselyä tehdessä.



KUVA 1. Havaintopaikan tietojen HTML-lomake

Tutkittiin palvelimen palauttamien tuloksien lähdekoodia ja varsinaisten tulosten kehyksen sivun osoitetta ja todettiin, että HTTP-kyselyssä lähetyt muuttujat ovat nimeltään Method, txtSelected, txtSelHeader, txtCalculation, txtCalculation-Name, txtGraph, txtAltitude, txtQFore, txtIceThick, txtTimeperiod, txtTimefilter, txtCalcTime, txtFreqType ja txtFreqDistribution (7).

Moni näistä muuttujista lähetetään palvelimelle ilman arvoa eli tyhjänä. Muuttuja Method voidaan pitää kiinteästi arvossa PrintToBrowser, jolloin palvelu palauttaa tiedot HTML-muodossa, jonka käsittelyyn PHP tarjoaa valmiit funktiot. Muuttuja txtSelHeader voidaan pitää kiinteästi arvossa #STATIONNUMBER,#NAME,#PARAM, eli kerrotaan palvelimelle, että halutaan tiedot Tunnus,Nimi,Suure. Muuttuja txtGraph on kiinteä 1. Muuttuja txtSelected on ns. OIVA ID, joka lähetetään pavelimelle. Muuttuja txtAltitude on korkeustaso, joka voi vaihdella melontapaikan mukaan. Muuttuja txtTimePeriod on alkupäivämäärä-loppupäivämääräarvoinen ja päivämäärät ovat muotoa pp.kk.vvvv. Kun syötetään muuttujan ja arvot kyselyn URL:ään, huomataan, että palvelin hyväksyy HTTP GET -kyselyt, mikä helpottaa testaamista jonkin verran.

Kun selaimen evästetiedot tyhjennettiin, huomattiin, että palvelu ei palauta tuloksia normaaliin tapaan, vaan esittää tulossivun sijaan kirjautumissivun. Kirjautumistietojen siirtämistä tarkkailtiin Wireshark-sovelluksella, jolla saatiin selvitettyä, että HTTP-tietoliikenteen HEADER-tiedoissa siirtyi evästetietoja, jotka tarvitaan sivuston käyttämiseen. Evästetietoja tarkkailemalla ja poistamalla saatiin myös selville, että olennainen evästetieto on OivaUserName ja ns. ASP SESSION ID -evästeet eivät ole pakollisia siirrettäviä. Lisäksi huomattiin, että OivaUserNamen arvo on sama kuin OIVA käyttäjätunnus, paitsi merkit ”@” ja ”.” on korvattu tekstillä ”%5F”, mikä vastaa URL-enkoodausta merkille ”_” (9).

4.2.2 OIVA-tietojen haun toteuttaminen

OIVA-kyselyjen luomista varten toteutettiin funktio buildOivaQuery (kuva 2). Koska ainoat muuttuvat lähetettävät tiedot ovat txtSelected, txtAltitude ja txtTimeperiod, mutta muutkin kentät on lähetettävä, niin oli järkevää tehdä funktio, joka ottaa parametreinaan vain OIVA ID:n ja Korkeusarvon, laskee halutut päivämäärät ja palauttaa kyselyssä käytettävän kysely-resurssin. (Kuva 3.)

```
public static function buildOivaQuery($oivaId,$altitudeValue = '-') {
    $params = &JComponentHelper::getParams( 'com_riverinfo' );
    $days = $params->get('maxAge');
    $timePeriod=date("d.m.Y",time() - ($days * 24 * 60 * 60)).'-'.date("d.m.Y");
    $array = array( 'Method' => 'PrintToBrowser',
        'txtSelected' => $oivaId,
        'txtSelHeader' => '#STATIONNUMBER,#NAME,#PARAM',
        'txtCalculation' => '',
        'txtCalculationName' => '',
        'txtGraph' => '1',
        'txtAltitude' => $altitudeValue,
        'txtRain' => '',
        'txtQFore' => '',
        'txtIceThick' => '',
        'txtTimeperiod' => $timePeriod,
        'txtTimefilter' => '',
        'txtCalcTime' => '',
        'txtFreqType' => '',
        'txtFreqDistribution' => '',
        );
    return http_build_query($array);
}
```

KUVA 2. Funktio kyselyresurssin luomiseen

```
$data = self::buildOivaQuery($item->oivaId,$item->altitudeValue);
```

KUVA 3. Sijoitetaan buildOivaQueryn luoma resurssi $data-muuttujaan

Koska evästetiedoissa tarvitsee olla vain käyttäjänimen pohjalta luotu OivaUserName-arvo eikä sessio-pohjaista ID:tä tarvita, voidaan evästetiedot pitää komponentin asetukset-sivun asetuksena eikä varsinaista sisäänkirjautumista tarvitse tehdä itse komponentin kautta. Evästetiedot voidaan ladata muuttujaan JComponentHelper-luokan getParams-funktion avustuksella. (Kuva 4.)

```
$params = &JComponentHelper::getParams( 'com_riverinfo' );
$url = $params->get('oivaUrl');
$cookie = $params->get('oivaCookie');
```

KUVA 4. OIVA-palvelun URL ja evästetiedot ladataan muuttujiin

HTTP-tietoliikenteen kautta tapahtuvaa tiedonhakua varten toteutettiin doRequest-funktio, joka lähettää HTTP GET -kyselyn ja sisällyttää evästetiedot HTTP-kyselyn HEADER-tiedoissa ja palauttaa vastaanotetun sivun merkkijonona. (Kuva 6.)

HTTP-kyselyiden toteutuksessa käytetään PHP:n fopen-funktiota (kuva 5), joka on tiedostojen avaamiseen tarkoitettu funktio. Funktio tukee useita protokollia, joten sen avulla voi myös hakea tietoja HTTP-protokollan yli.

```
public static function doRequest($url, $data, $optional_headers = null, $post = false) {
    if ($post) {
        $params = array('http' => array(
        'method' => 'POST',
        'content' => $data
        ));
    }
    if ($optional_headers !== null) {
        $params['http']['header'] = $optional_headers;
    }
    $ctx = stream_context_create($params);
    stream_context_set_params($ctx,
        array("notification" => "stream_notification_callback"));

    $fp = @fopen($url."?".$data, 'rb', false, $ctx);
```

KUVA 5. Lisätään $optional_headersin sisältämät evästetiedot HEADERiin ja avataan sivu

```
$response = @stream_get_contents($fp);
if ($response === false) {
    throw new Exception("Problem reading data from $url, $php_errormsg");
}
return $response;
```

KUVA 6. Funktio palauttaa sivun sisällön

4.2.3 OIVA-tietojen parseroiminen

Palvelimelta tullut palaute otettiin selaimelle tulostettavassa muodossa asettamalla Method arvoon PrintToBrowser, jolloin palaute oli HTML-muodossa. Työn kannalta kiinnostavat arvot ovat mittauspäivämäärä ja vedenkorkeus- tai virtaamatieto. Arvot on esitetty HTML-taulukkona (8), joten käymällä läpi palautteen TD-elementit (kuva 7 ja kuva 8) voidaan helposti parseroida tiedot. Elementtejä tarkistetaan kuluvasta päivämäärästä menneisyyteen päin etsien päivämäärämerkintää. Jos ajalle löytyy merkintä, palautetaan päivämäärä ja arvo nimikoidussa taulukossa.

```
public static function getHeight($str,$date) {
    $DOM = new DOMDocument();
    $DOM->loadHTML($str);

    //get all TD
    $items = $DOM->getElementsByTagName('td');
```

KUVA 7. Ladataan TD-elementtien tiedot muuttujaan

```
$params = &JComponentHelper::getParams( 'com_riverinfo' );
$days = $params->get('maxAge');

//display all TD text
for ($dateOffset = 0; $dateOffset <= $days ; $dateOffset++) {
    for ($i = 0; $i < $items->length; $i++) {
        if ($items->item($i)->nodeValue == date(
                                            "j.n.Y",
                                            $date -($dateOffset*24*60*60)
                                            ))
            {
            $height = $items->item($i + 1)->nodeValue;
            $result = array(
                    'date' => date("Y-m-d",$date -($dateOffset*24*60*60)),
                    'height' => $height
                );
            return $result;
```

KUVA 8. Etsitään tiedot sopivalle päivämäärälle ja palautetaan tiedot

4.2.4 OIVA-tietojen päivitys

OIVA-tietojen ajoittaista päivitystä varten toteutettiin päivityssivu, joka käy tarkistamassa OIVA-palvelusta uusimmat tiedot, mikäli niitä ei ole päivätty kuluvalle päivälle ja tietojen viimeisimmästä päivitysyrityksestä on kulunut yli asetuksissa määritetty aika. Päivityssivun ei ole määrä olla mikään käyttäjälle varsinaisesti näkyvissä oleva sivu, vaan sivu, jonka lataaminen suoritetaan automaattisesti tietyin väliajoin, jolloin saadaan toteutettua tietojen automaattinen päivitys.

Päivityssivua varten toteutettiin päivitys-funktio, joka lataa kannasta melontapaikkojen tiedot melontapaikoille, joiden viimeisin päivitysyritys on tapahtunut yli asetuksissa määritetyn tuntimäärän. Jokaiselle löydetylle melontapaikalle koetetaan asettaa ”päivityslukko” ja tarkistetaan, saatiinko lukko varmasti asetettua. Päivityslukon tarkoitus on varmistaa, ettei monta erillistä yhtäaikaista instanssia koeta hakea tietoja OIVA-palvelusta yhtä aikaa. Jos päivityslukon asetus on onnistunut ja lukon arvo vastaa suoritettavan instanssin lukko-arvoa, tarkastetaan

uusimmat merkinnät OIVA-palvelusta ja talletetaan uusin merkintä melontapaikan tietoihin.

4.2.5 OIVA-evästetietojen tallennus

Sivuston ylläpitäjä voi päättää, haluaako syöttää komponentin asetuksiin evästetiedot vaiko käyttäjätunnuksen ja salasanan. Mikäli ylläpitäjä syöttää evästetiedot, ei käyttäjätunnusta ja salasanaa tarvita komponentin toiminnan kannalta. Ylläpitäjä voi halutessaan myös syöttää asetuksiin käyttäjänimen ja salasanan, jolloin komponentti itse kirjautuu OIVA-palveluun ja tallettaa asetuksiinsa tarvittavat evästeet.

OIVA-palveluun kirjautumista varten luotiin login-funktio (kuva 9). Funktio hakee evästetiedon, käyttäjänimen ja salasanan asetuksista. Jos evästetietoa ei ole asetuksissa, luo funktio HTTP-kyselyn ja lähettää kirjautumiskyselyn käyttäen doRequest-funktiota.

```
public static function login($showMessages = false)
{
    $params = &JComponentHelper::getParams( 'com_riverinfo' );
    $url = $params->get('oivaLoginUrl');
    $username = $params->get('oivaUsername');
    $password = $params->get('oivaPassword');
    $cookie = $params->get('oivaCookie');
    if ($cookie) return;
    $data = http_build_query( array(
                'userName'  => $username,
                'password' => $password
            )
            );
    $result = self::doRequest($url, $data,null,true);
```

KUVA 9. OIVA-palveluun sisäänkirjautumisen funktio

Evästetietojen tallentamista varten toteutettiin setCookie-funktio (kuva 10), joka tallentaa evästetiedot tietokantaan. Lisättiin doRequest-funktion sisälle evästetietotarkistus ja kutsu setCookie-funktioon (kuva 11).

```
public static function setCookie($cookie)
{
    $params = JComponentHelper::getParams('com_riverinfo');
    $params->set('oivaCookie', $cookie);
    $db = JFactory::getDBO();
    $query = $db->getQuery(true);
    $query->update('#__extensions AS a');
    $query->set('a.params = ' . $db->quote((string)$params));
    $query->where('a.element = "com_riverinfo"');
    $db->setQuery($query);
    $db->query();
```

KUVA 10. Funktio tallentaa evästetiedot tietokantaan

```
$meta = stream_get_meta_data($fp);

$params = &JComponentHelper::getParams( 'com_riverinfo' );
$cookie = $params->get('oivaCookie');
if (!$cookie) {
    foreach($meta['wrapper_data'] as $i => $item):
        if (preg_match('/Set-Cookie: (OivaUserName=[^;]+);/',
                        $item,$matches)) {
        self::setCookie( $matches[1] );
        }
    endforeach;
```

KUVA 11. Tarkistetaan, onko evästetietoja, ja jos uusi evästetieto löytyy, kutsutaan setCookie-funktiota

4.3 Tietokanta

Tietokantayhteydet toteutettiin käyttäen Joomlan tarjoamia tietokantarajapintoja. Joomlan tarjoamat tietokantarajapinnat helpottavat ja nopeuttavat tietokantayhteyksien toteutusta, ja samalla komponentti osaa käyttää Joomlan asennuksen aikana syötettyjä käyttäjätunnuksia, salasanoja ja tietokanta-asetuksia.

Tietokantayhteys muodostetaan JDatabase-luokan oliolla. Olio luodaan käyttämällä JFactory-luokan getDBO-funktiota, joka palauttaa olion. Tietokantakysely luodaan JDatabaseQuery-luokan olioon. (Kuva 12.)

```
$db = JFactory::getDBO();
$query = $db->getQuery(true);
$query->select('#__riverinfo_rivers.id as id ,name,
                description,#__categories.title as category,
                catId, oivaId, minHeight, maxHeight');
$query->from('#__riverinfo_rivers');
$query->leftJoin('#__categories on catid=#__categories.id');
$db->setQuery((string)$query);
$rivers = $db->loadObjectList();
```

KUVA 12. Haetaan tietokannasta melontapaikat $rivers-olioon

4.4 Lokalisaatio

Läpi koko komponentin kehityksen komponentti toteutettiin käyttäen lokalisaatiotukea. Jokainen käyttäjälle tai ylläpitäjälle esitettävä tekstirivi on siis käännettävissä mille kielelle tahansa ja näytetään asetuksissa valitulla kielellä, jos käännös kielelle on tehty.

Kielimääritelmät ovat erikseen sivu- ja ylläpito-osioille osioiden language-kansion sisällä. Kielimääritelmätiedostoihin asetetaan yksi rivi käännöstietoa kohti, joka sisältää ”avaimen”, jossa on etuliitteenä COM_KOMPONENTIN_NIMI, eli tämän työn tapauksessa COM_RIVERINFO, eikä se saa sisältää välejä tai useimpia erikoismerkkejä. Avaimen jälkeen on yhtäsuuruusmerkki ja lainausmerkeissä annettu käännetty teksti. (Kuva 13.)

```
COM_RIVERINFO_RIVERINFO_DETAILS="River details"
```

KUVA 13. Englanninkielinen kielimäärittely melontapaikan tarkennetuille tiedoille

Koodissa kielimääritelmien käyttäminen on yksinkertaista. Kielimääritelmää käytetään JText-luokan _-funktiolla. (Kuva 14.)

```
<?php echo JText::_( 'COM_RIVERINFO_RIVERINFO_DETAILS' ); ?>
```

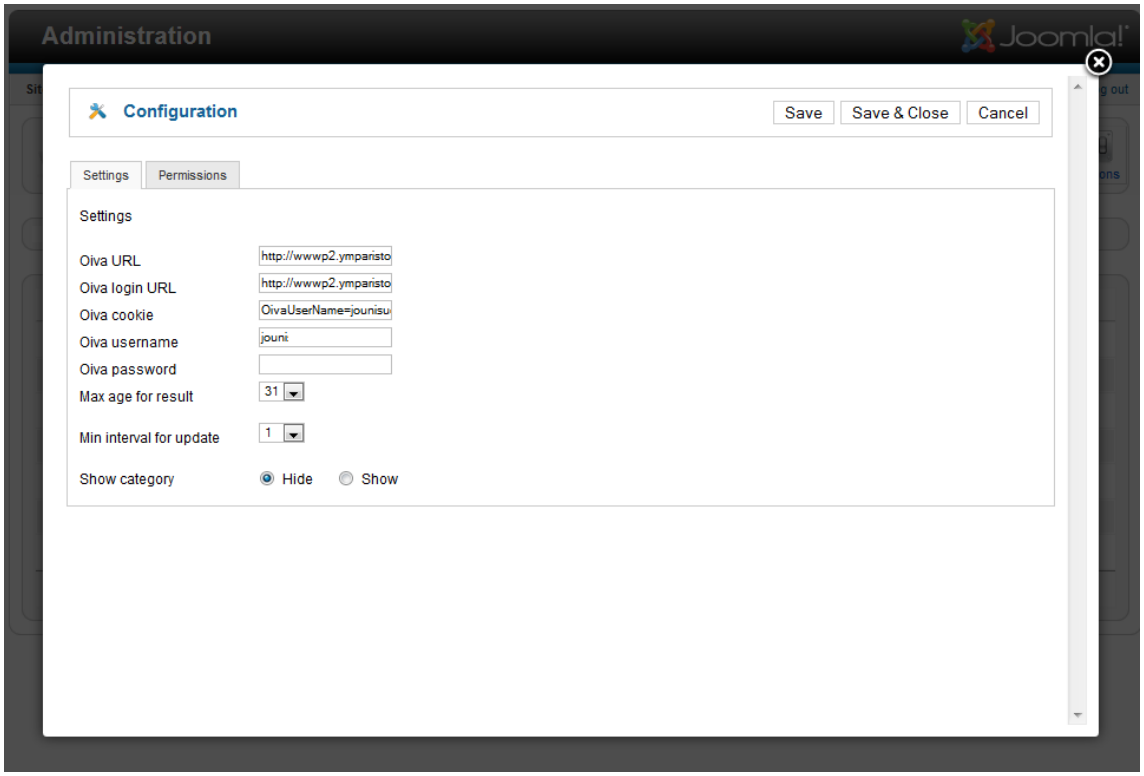
KUVA 14. Kielimääritelmän käyttäminen koodissa

4.5 Ylläpito-osa

Sivustoon toteutettiin myös komponentin ylläpito-osa.

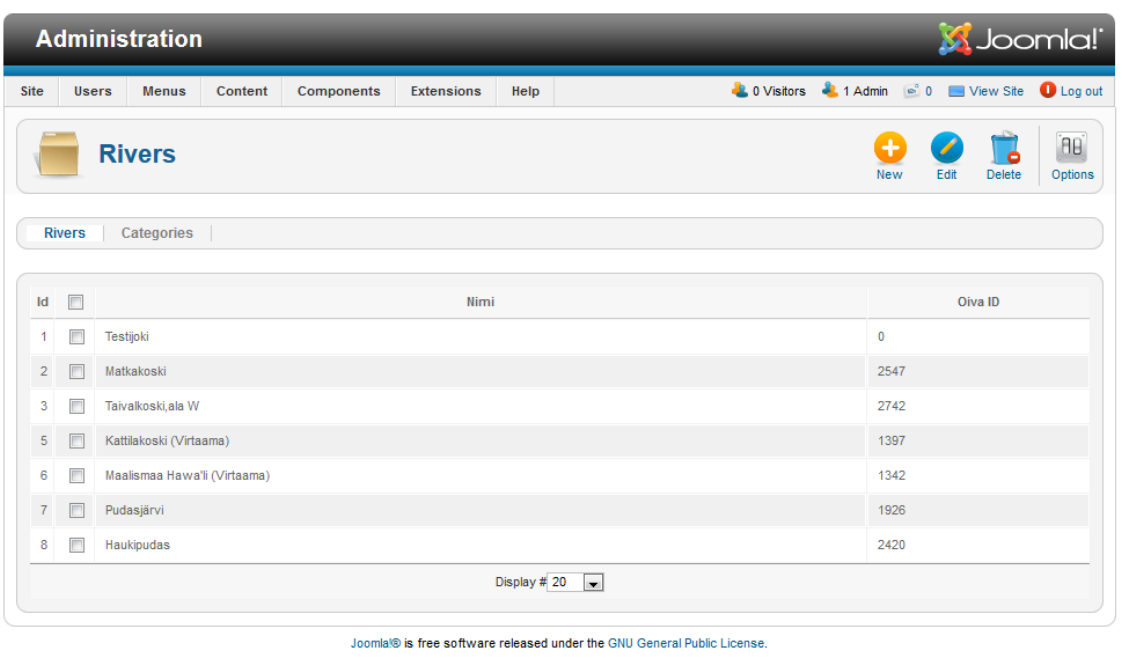
4.5.1 Ylläpito-osan ominaisuudet

Komponentin ylläpitoasetuksissa pääsee lisäämään, poistamaan ja muokkaamaan melontapaikkoja. Lisäksi siellä asetetaan komponentin toiminnan kannalta oleelliset asetukset. Toiminnan kannalta oleellisia asetuksia ovat OIVA-palvelun URL-osoite, johon vedenkorkeuskyselyt lähetetään, sekä HTTP-liikenteen evästetiedot. Lisäksi asetuksissa pääsee valitsemaan näytetäänkö melontapaikkojen kategoriat vedenkorkeusnäkymässä. Tämän lisäksi voidaan asettaa OIVA-palvelun tunnus ja salasana. Myös OIVA-palvelun sisäänkirjautumistoiminnon URL:lle on varattu oma kenttänsä. (Kuva 15.)



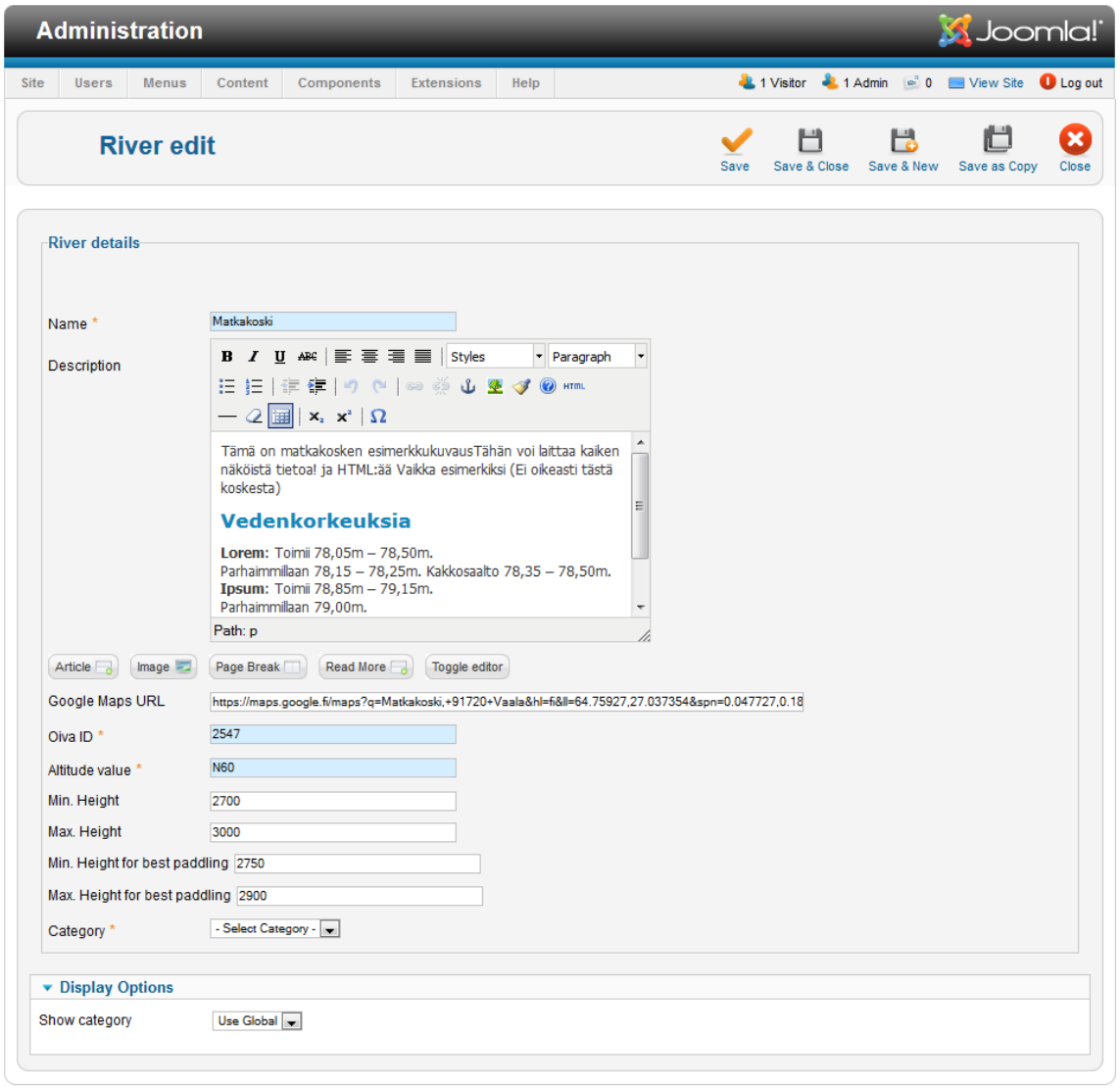
KUVA 15. Komponentin asetukset

Ylläpito-osan vakionäkymä listaa tietokantaan syötetyt melontapaikat, joita pääsee muokkaamaan valitsemalla melontapaikan ja painamalla Muokkaa-ikonia (kuva 16).



KUVA 16. Melontapaikat listattuna ylläpitonäkymässä

Melontapaikan muokkausnäkymässä pääsee nimeämään melontapaikan, antamaan melontapaikalle vapaamuotoisen kuvauksen, syöttämään URL:n Google Maps -sijaintiin sekä syöttämään OIVA-palvelun käyttämän ID-numeron vedenkorkeudelle tai virtaamalle (kuva 17). Melontapaikalle täytyy myös määrittää minimi- ja maksimiarvo melontakelpoiselle vedenkorkeudelle, jotta komponentti voi todeta OIVA-palvelusta haetut vedenkorkeudet melontakelpoisiksi. Lisäksi näkymässä pääsee määrittämään korkeusarvon, jos se on melontapaikan kohdalla tarpeen. On myös mahdollista määrittää melontapaikalle kategoria, mutta se todennäköisesti jäänee tyhjäksi seurattavien melontapaikkojen vähäisyyden takia.



KUVA 17. Melontapaikan muokkausnäkymä

4.5.2 Ylläpito-osan toteutus

Komponentin asetuksia varten tehty näkymän toteutus on todella yksinkertainen. Asetuksia varten täytyi luoda komponentin admin-kansioon config.xml-tiedosto, jossa määritetään komponentin käyttämät asetukset ja niiden tyypit. (Kuva 18, kuva 19, kuva 20 ja kuva 21.) Jokaiselle asetukselle annettiin nimi, tyyppi, nimike ja kuvaus. Lisäksi joillekin tyypeille pystyy määrittämään erinäisiä arvoja. Rajattiin päivitystiheys-arvot välillä 1–24 tuntia ja vanhin OIVA-palvelusta hyväksyttävä tieto välille 1–61 vuorokautta (kuva 19). Koska komponentti kehitettiin tukemaan useita kieliä, annettiin nimikkeelle ja kuvaukselle kielimääritelmäavain, jotta nimike ja kuvaus voidaan esittää ylläpitäjälle halutulla kielellä.

```
<?xml version="1.0" encoding="utf-8"?>
<config>
	<fieldset
		name="names"
		label="COM_RIVERINFO_CONFIG_NAME_SETTINGS_LABEL"
		description="COM_RIVERINFO_CONFIG_NAME_SETTINGS_DESC"
	>
		<field
			name="oivaUrl"
			type="text"
			label="COM_RIVERINFO_RIVERINFO_FIELD_OIVAURL_LABEL"
			description="COM_RIVERINFO_RIVERINFO_FIELD_OIVAURL_DESC"
			default=""
		/>
```

KUVA 18. Määritetään komponentin OIVA-kyselyyn käytettävä URL-asetus

```
<field
	name="maxAge"
	type="integer"
	label="COM_RIVERINFO_RIVERINFO_FIELD_MAXAGE_LABEL"
	description="COM_RIVERINFO_RIVERINFO_FIELD_MAXAGE_DESC"
	default="31"
	first="1"
	last="61"
	step="1"
/>
<field
	name="minInterval"
	type="integer"
	label="COM_RIVERINFO_RIVERINFO_FIELD_MININTERVAL_LABEL"
	description="COM_RIVERINFO_RIVERINFO_FIELD_MININTERVAL_DESC"
	default="6"
	first="1"
	last="24"
	step="1"
/>
```

KUVA 19. Asetuksissa määrätään vanhin hyväksytty OIVAn antama arvo ja päivitystahti

```
<field
	name="show_category"
	type="radio"
	label="COM_RIVERINFO_RIVERINFO_FIELD_SHOW_CATEGORY_LABEL"
	description="COM_RIVERINFO_RIVERINFO_FIELD_SHOW_CATEGORY_DESC"
	default="0"
>
	<option value="0">JHIDE</option>
	<option value="1">JSHOW</option>
</field>
```

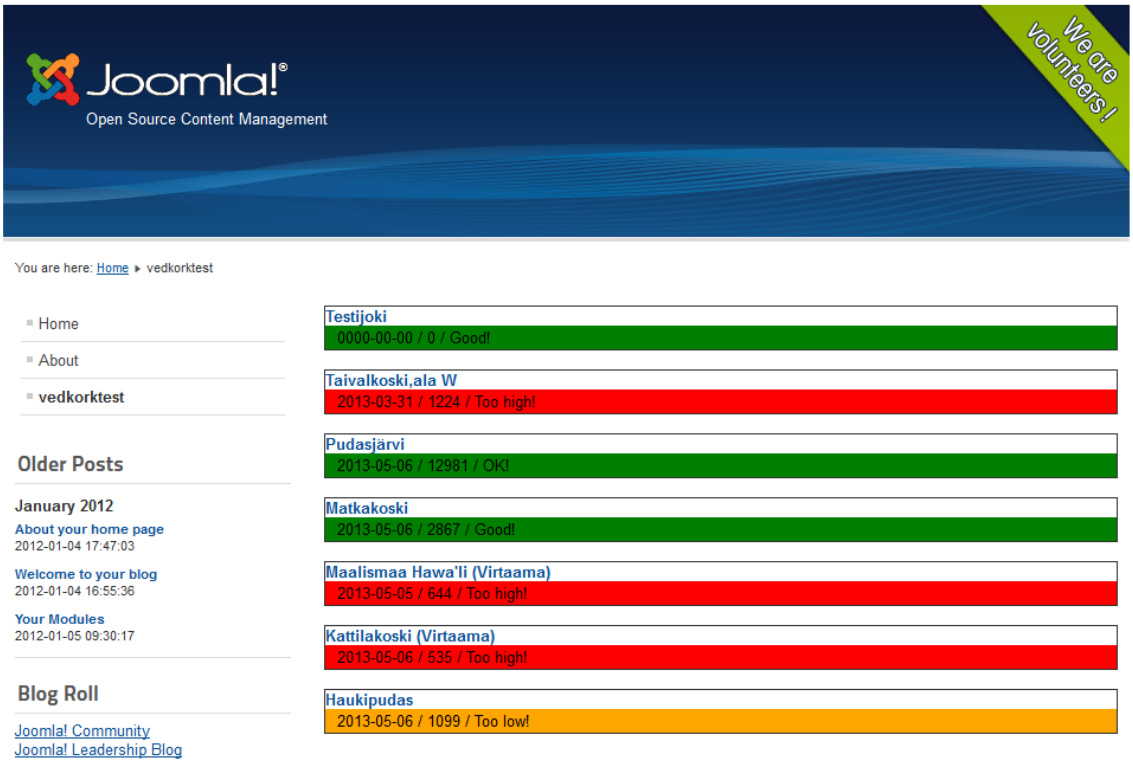
KUVA 20. Määritetään asetus ja vaihtoehdot kategorian näyttämiselle

```
        <fieldset
                name="permissions"
                label="JCONFIG_PERMISSIONS_LABEL"
                description="JCONFIG_PERMISSIONS_DESC"
        >
                <field
                        name="rules"
                        type="rules"
                        label="JCONFIG_PERMISSIONS_LABEL"
                        class="inputbox"
                        validate="rules"
                        filter="rules"
                        component="com_riverinfo"
                        section="component"
                />
        </fieldset>
</config>
```

KUVA 21. Liitetään komponenttiin Joomlan valmiit sääntöasetukset

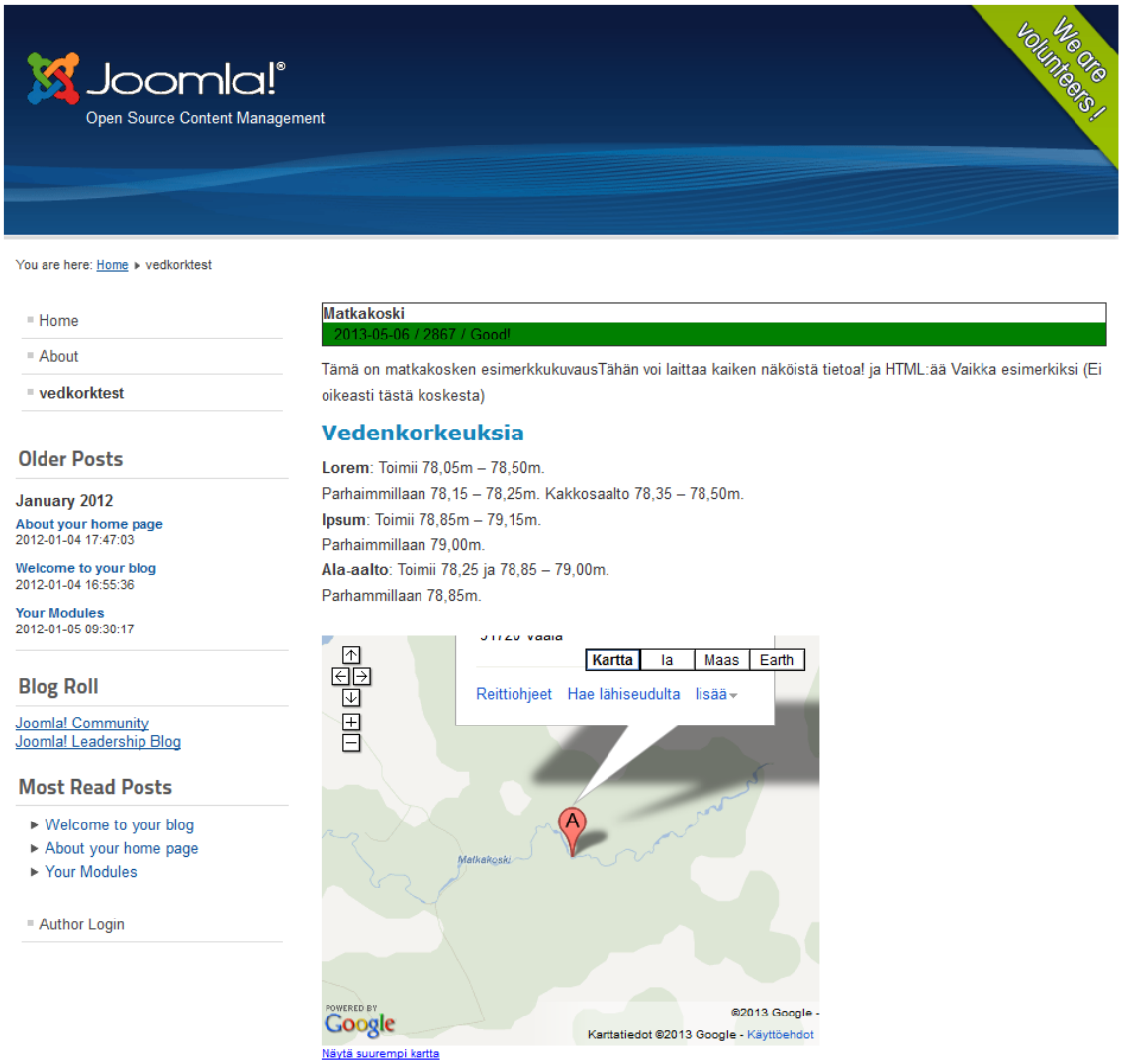
4.6 Vedenkorkeudet-osa

Vedenkorkeudet-osa kattaa käyttäjälle näkyvät sivut, joita on kaksi. Käyttäjälle ensimmäisenä näkyvä melontapaikat-näkymä listaa kaikki melontapaikat vedenkorkeuksineen ja mittauspäivämäärineen (kuva 22). Melontapaikan korkeuden taustaväri vaihtelee sen mukaan, miten vedenkorkeus soveltuu melontaan. Soveltuvuustasoja on neljä. Tasot ovat liian alhainen vesimäärä, liian korkea vesimäärä, kelvollinen vesimäärä ja melomiseen parhaiten sopiva vesimäärä. Kelvollinen vesimäärä tarkoittaa, että melominen on mahdollista, mutta vettä olisi mielellään vähän enemmän tai vähemmän. Korkeuden kelpoisuus kerrotaan myös tekstimuodossa, jotta käyttäjä paremmin sisäistää taustavärien merkityksen.



KUVA 22. Melontapaikat-näkymä listaa melontapaikat

Painamalla melontapaikan nimeä voidaan siirtyä melontapaikkanäkymään (kuva 23), joka näyttää kyseisen melontapaikan tiedot hieman tarkemmin kuin melontapaikat-näkymässä. Melontapaikkanäkymässä kerrotaan kyseessä olevan melontapaikan kohdalta kaikki tiedot, jotka näkyi melontapaikat-näkymässäkin, mutta niiden lisäksi näytetään melontapaikalle syötetty kuvaus ja sijainti kartalla, mikäli Google Maps -sijainti-URL on syötetty melontapaikan tietoihin.



KUVA 23. Melontapaikka-näkymä näyttää melontapaikan tiedot syvällisemmin

5 YHTEENVETO

Työn tavoitteena oli luoda Joomla-sisällönhallintajärjestelmässä toimiva komponentti joka esittää sivuston käyttäjälle selkeästi melontapaikan melontakelpoisuuden. Komponentin avulla melojalta jää turha tulkinnanvaraisuus pois melontapaikan melontakelpoisuudesta, niin kuin tavoitteena oli.

PHP-kehittäminen oli ennestään tuttua ja hyvin hallussa, mutta Joomla-kehittämisestä ei ollut aiempaa kokemusta. Joomla oli kuitenkin tuttu sivuston ylläpitäjän näkökulmasta. Joomla-komponenttien kehittämisessä alkuun pääseminen oli jokseenkin hidasta, mutta alkukangertelun jälkeen huomattiin, että Joomla tarjoaa kehittäjälle paljon käteviä rajapintoja, ja jotkin asiat olivat paljon helpompia kehittää, kuin oli uskaltanut kuvitellakaan. Ylläpitoon tarkoitettujen asetussivujen luominen oli Joomla-kehittämiseen kunnolla tutustumisen jälkeen hyvin helppoa. Määritettävät kentät sai asettaa XML-muotoisessa tiedostossa, ja eri ylläpitosivuissa varsinaisen PHP-koodin käyttö jää hyvin vähälle. Joomlan tietokantakyselyitä varten tarjotut toiminnot yksinkertaistavat tietokantojen käyttämistä huomattavasti PHP:n itsensä tarjoamiin tietokantaluokkiin verrattuna, mikä toki selittyy sillä, että Joomla-komponentin ei tarvitse huolehtia tietokantapalvelimen osoitteesta, käyttäjätunnuksesta, salasanasta tai edes tietokannasta itsestään, vaan tieto taulusta ja tietueista riittää.

Komponenttiin saatiin opinnäytetyössä toteutettua kaikki toiminnallisuus, mitä toimeksiantaja oli komponenttiin toivonut, ja toteutus pysyi sopivan yksinkertaisena. Yksinkertainen toteutus oli yksi tavoitteista, koska komponentti oli määrä toteuttaa suhteellisen lyhyessä aikataulussa lisäksi yksinkertainen toteutus helpottaa komponentin kehitystä jatkossa.

Komponentti on toteutettu toimeksiantajan tarpeisiin kohdistetusti. Jos komponenttia jaettaisiin tai markkinoitaisiin muihin melontaseuroihin tai jos toimeksiantajana toiminut seura haluaisi listata melontapaikat koko Suomesta, voisi lisätoiminnoille tulla tarvetta. Jos melontapaikkoja olisi suuri määrä, voisi olla tarpeen rajata listattavia melontapaikkoja esimerkiksi läänin mukaan. Melontapaikkojen

lisääminen on tällä hetkellä hieman työlästä ja sitä varten käyttäjän on itse selattava ympäristöhallinnon OIVA-palvelua sen sivuilla, eikä uusien melontapaikkojen OIVA ID-numeroita ja nimiä voi hakea suoraan komponentin sisältä. Komponentin sisälle rakennettu melontapaikkojen ID-numeroiden, nimien ja asetusten hakeminen olisi työläs toteuttaa, ja tämänhetkisen tarpeen kannalta se ei ole kannattavaa.

Opinnäytetyötä tehdessä opin paljon Joomlan tarjoamista eduista ja komponenttien kehittämisestä Joomla-sisällönhallintajärjestelmälle. Uskon kokemuksen olevan tulevaisuudessa hyödyksi, sillä Joomla on tällä hetkellä todella laajassa käytössä. Opinnäytetyössä tehdyn komponentin toteuttaminen antaa ainakin hyvän kokemuspohjan, jos joskus haluan alkaa tehdä maksullista Joomla-komponenttia johonkin tarkoitukseen.

LÄHTEET

1. PHP: What is PHP? 2013. Saatavissa: http://www.php.net/manual/en/intro-whatis.php. Hakupäivä 28.4.2013.

2. PHP: What can PHP do? 2013. Saatavissa: http://www.php.net/manual/en/intro-whatcando.php. Hakupäivä 28.4.2013.

3. MySQL :: MySQL 5.6 Reference Manual :: 1.3.1 What is MySQL? 2013. Saatavissa: http://dev.mysql.com/doc/refman/5.6/en/what-is-mysql.html. Hakupäivä 28.4.2013.

4. MySQL :: MySQL Customers. Saatavissa: http://www.mysql.com/customers/. Hakupäivä 28.4.2013.

5. Joomla? 2011. Saatavissa: http://www.joomla.fi/mika-on-joomla. Hakupäivä 28.4.2013.

6. Technical Requirements. Saatavissa: http://www.joomla.org/technical-requirements.html. Hakupäivä 24.5.2013.

7. Havaintopaikan tiedot. Saatavissa: http://wwwp2.ymparisto.fi/scripts/palvelut.asp, Ympäristötiedon hallintajärjestelmä Hertta, Vesivarat, Hydrologiset havainnot, Vedenkorkeus, Matkakoski, Etsi, Taulukko. Vaatii kirjautumisen. Hakupäivä 28.4.2013.

8. Tulokset näytöllä. Saatavissa: http://wwwp2.ymparisto.fi/scripts/palvelut.asp, Ympäristötiedon hallintajärjestelmä Hertta, Vesivarat, Hydrologiset havainnot, Vedenkorkeus, Matkakoski, Etsi, Taulukko, Näytölle. Vaatii kirjautumisen. Hakupäivä 28.4.2013.

9. HTML URL Encoding Reference. Saatavissa: http://www.w3schools.com/tags/ref_urlencode.asp. Hakupäivä 6.5.2013.